CHICAGO JOURNAL OF THEORETICAL COMPUTER SCIENCE 2014, Article 08, pages 1–16
http://cjtcs.cs.uchicago.edu/

Computing in Matrix Groups Without Memory

Peter J. Cameron Ben Fairbairn Maximilien Gadouleau*

Received November 5, 2013; Revised August 27, 2014 and in final form September 29, 2014; Published November 2, 2014

Abstract: Memoryless computation is a novel means of computing any function of a set of registers by updating one register at a time while using no memory. We aim to emulate how computations are performed on modern cores, since they typically involve updates of single registers. The computation model of memoryless computation can be fully expressed in terms of transformation semigroups, or in the case of bijective functions, permutation groups. In this paper, we view registers as elements of a finite field and we compute linear permutations without memory. We first determine the maximum complexity of a linear function when only linear instructions are allowed. We also determine which linear functions are hardest to compute when the field in question is the binary field and the number of registers is even. Secondly, we investigate some matrix groups, thus showing that the special linear group is internally computable but not fast. Thirdly, we determine the smallest set of instructions required to generate the special and general linear groups. These results are important for memoryless computation, for they show that linear functions can be computed very fast or that very few instructions are needed to compute any linear function. They thus indicate new advantages of using memoryless computation.

*Supported by EPSRC grant EP/K033956/1.

Key words and phrases: memoryless computation, linear functions, matrix groups, general linear group, special linear group, generating sets, sequential updates

DOI: 10.4086/cjtcs.2014.008

1 Introduction

1.1 Memoryless computation

Typically, swapping the contents of two variables x and y requires a buffer t, and proceeds as follows (using pseudo-code):

$$\begin{aligned} t &\leftarrow x \\ x &\leftarrow y \\ y &\leftarrow t. \end{aligned}$$

However, the famous XOR swap (when x and y are sequences of bits), which we view in general as addition over a vector space:

$$\begin{aligned} x &\leftarrow x+y \\ y &\leftarrow x-y \\ x &\leftarrow x-y, \end{aligned}$$

performs the swap without any use of memory.

While the example described above (the so-called XOR swap) is folklore in Computer Science, the idea to compute functions without memory was developed in [2, 3, 4, 6, 7, 8, 5] and surveyed in [5] and extended in [12]. Amongst the results derived in the literature is the non-trivial fact that any function of n-bit input and n-bit ouput can be computed using memoryless computation. Moreover, only a number of updates linear in the number of registers is needed: any function of n variables can be computed in at most $4n-3$ updates [5, 12], and only $2n-1$ updates if the function is bijective [4].

Memoryless computation has the potential to speed up computations not only by avoiding time-consuming communication with the memory but also by effectively combining the values contained in registers. This indicates that memoryless computation can be viewed as an analogue in computing to network coding [1, 17], an alternative to routing on networks. It is then shown in [12] that for certain manipulations of registers, memoryless computation uses arbitrarily fewer updates than traditional, "black-box" computing.

As unveiled in the proof of the founding theorem of memoryless computation given in [12], memoryless computation can be best approached by algebraic methods, such as transformation semigroups or permutation groups, when considering bijective functions. Memoryless computation is then a unique area of theoretical computer science, which brings new insights and possible applications to some well-known results in algebra.

1.2 Model for computing in matrix groups without memory

In this paper, we are interested in computing linear bijective functions without memory. Some results already appear in the literature about these functions. For instance, any linear function over can be computed in at most $2n-1$ updates [5] for large classes of rings; in this paper, we lower that upper bound to $\lfloor 3n/2 \rfloor$ for finite fields, which is tight. The number of updates required to compute any

manipulation of variables (i.e., any function of the form $f(x_1,\ldots,x_n) = (x_{1\phi},\ldots,x_{n\phi})$ for some function $\phi : \{1,\ldots,n\} \to \{1,\ldots,n\}$) is also determined in [12].

Foremost, let us recall some notation and results from the literature. Let $A := \mathrm{GF}(q)$ be a finite field (the *alphabet*) and $n \geq 2$ be an integer representing the number of registers (also called variables) $x_1,\ldots,x_n$. We denote $[n] = \{1,2,\ldots,n\}$. The elements of A^n are referred to as *states*, and any state $a \in A^n$ is expressed as $a = (a_1,\ldots,a_n)$. For any $1 \leq k \leq n$, the k-th unit state is given by $e^k = (0,\ldots,0,1,0,\ldots,0)$ where the 1 appears in coordinate k. We also denote the all-zero state as e^0.

For any $f \in \mathrm{Sym}(A^n)$, we denote its n coordinate functions as $f_1,\ldots,f_n : A^n \to A$, i.e. $f(x) = (f_1(x),\ldots,f_n(x))$ for all $x = (x_1,\ldots,x_n) \in A^n$. We say that the i-th coordinate function is *trivial* if it coincides with that of the identity: $f_i(x) = x_i$; it is nontrivial otherwise.

A bijective *instruction* is a permutation g of A^n with one nontrivial coordinate function:

$$g(x) = (x_1,\ldots,x_{j-1},g_j(x),x_{j+1},\ldots,x_n)$$

for some $1 \leq j \leq n$. We say the instruction g *updates* the j-th coordinate. We can represent this instruction as

$$y_j \leftarrow g_j(y)$$

where $y = (y_1,\ldots,y_n) \in A^n$ represents the contents of the registers. A *program* computing f is simply a sequence of instructions whose combination is f; the instructions are typically denoted one after the other.

With this notation, the swap of two variables can be viewed as computing the permutation f of A^2 defined as $f(x_1,x_2) = (x_2,x_1)$, and the program is given by

$$\begin{aligned} y_1 &\leftarrow y_1 + y_2 \qquad (= x_1 + x_2) \\ y_2 &\leftarrow y_1 - y_2 \qquad (= x_1) \\ y_1 &\leftarrow y_1 - y_2 \qquad (= x_2). \end{aligned}$$

In this paper, we want to compute a linear transformation $f : A^n \to A^n$, i.e.

$$f(x) = xM^\top$$

for some matrix $M \in A^{n\times n}$. We denote the rows of M as f_i. We restrict ourselves to linear instructions only, i.e. instructions of the form

$$y_i \leftarrow v \cdot y = \sum_{j=1}^{n} v_j y_j,$$

for some $v = (v_1,\ldots,v_n) \in A^n$. In particular, the instruction above is a permutation if and only if $v_i \neq 0$. Note that computing f without memory is then equivalent to computing M by starting from the identity matrix and updating one row at a time.

The set $\mathcal{M}(\mathrm{GF}(q)^n)$ of bijective linear instructions then corresponds to the set of nonsingular matrices with at most one nontrivial row: $\mathcal{M} = \{S(i,v) : 1 \leq i \leq n, v \in A^n(i)\}$, where

$$\begin{aligned} A^n(i) &= \{v \in A^n, v_i \neq 0\} \text{ for all } 1 \leq i \leq n, \\ S(i,v) &= \left(\begin{array}{c|c} I_{i-1} & 0 \\ \hline v & \\ \hline 0 & I_{n-i} \end{array}\right) \in A^{n\times n}. \end{aligned}$$

We remark that $S(i,v)^{-1} = S(i,-v_i^{-1}v)$ for all i,v.

Following [9], we say a permutation group G is *internally computable* if it can be generated by its instructions, i.e. if any element of the group can be computed by a program using instructions from G. For instance, Gaussian elimination proves that $\mathrm{GL}(n,q)$ is internally computable (swapping two rows, which corresponds to swapping two variables, is the only operation which cannot be viewed as an instruction; however the swap of two variables described in the beginning of the paper can be easily generalised to any alphabet). We prove in Proposition 3.1 that $\mathrm{SL}(n,q)$ is also internally computable. For any internally computable group G, two main problems arise. First, we want to know how fast we can compute any element of G: we will prove that the maximum complexity in the general linear group is $\lfloor 3n/2 \rfloor$ instructions in Theorem 2.1. More surprisingly, if $q = 2$ and n is even, then the matrices requiring $3n/2$ instructions are fully characterised in Proposition 2.2. Note that the average complexity over all elements of a group is also interesting; for $\mathrm{GL}(n,q)$, this quantity tends to n instructions when q is large [12].

Secondly, due to the large number of possible instructions, it seems preferable to work with restricted sets of instructions which could be efficiently used by a processor. Therefore, we also want to know the minimum number of instructions required to generate the whole group. We shall determine this for the special and general linear groups in Theorems 4.1 and 4.2, respectively. The fact that it is equal to n in most cases–and $n+1$ otherwise–shows how easy it is to compute linear functions without memory and how little space would be required to store those minimal sets of instructions.

For any internally computable group G and any $g \in G$, we denote the shortest length of a program computing g using only instructions from G as $\mathcal{L}(g,G)$; we refer to this quantity as the *complexity* of g in G. If $H \leq G$ and $\mathcal{L}(h,H) = \mathcal{L}(h,G)$ for all $h \in H$, we say that H is *fast* in G. It is still unknown whether $\mathrm{GL}(n,q)$ is fast in $\mathrm{Sym}(\mathrm{GF}(q)^n)$, i.e. if we cannot compute linear functions any faster by allowing non-linear instructions. However, we will prove in Proposition 3.1 that the special linear group is not fast in the general linear group (unless $q = 2$).

We would like to emphasize that we only consider bijective linear functions, i.e. computing in matrix groups. The case of any bijective function is studied in [9], where analogue results are derived for the symmetric and alternating groups of A^n (A being any finite set of cardinality at least 2).

The rest of the paper is organised as follows. In Section 2, we determine the maximum complexity of any matrix in $\mathrm{GL}(n,q)$ and investigate which matrices have highest complexity. Then, in Section 3, we determine whether some matrix groups are internally computable, and we show that $\mathrm{SL}(n,q)$ is internally computable but not fast in $\mathrm{GL}(n,q)$. Finally, in Section 4, we determine the minimum size of a generating set of instructions for both the special and general linear groups.

2 Maximum complexity in the general linear group

We first determine the maximum complexity of computing in $\mathrm{GL}(n,q)$. We would like to point out that the proof of Theorem 2.1 below uses different ideas to the one given in [5, Theorem 29], which builds programs of length $2n-1$ for a much wider class of rigs than finite fields.

Theorem 2.1. *Any matrix in* $\mathrm{GF}(q)^{n\times n}$ *can be computed in at most* $\lfloor 3n/2 \rfloor$ *linear instructions. This bound is tight and reached for some matrices in* $\mathrm{GL}(n,q)$.

Proof. We consider the general case where the matrix M we want to compute is not necessarily invertible. We prove the statement by induction on $n \geq 1$; it is clear for $n = 1$. Suppose it holds for up to $n-1$.

For any $S \subset [n]$, we refer to the matrix $M_S \in \mathrm{GF}(q)^{|S|\times|S|}$ with entries $M(i,j)$ for all $i,j \in S$ as the S-principal of M. Suppose that M has a nonsingular S-principal M_S, say $S = \{1,\ldots,k\}$ and express M as $M = \left(\begin{array}{c|c} M_S & N \\ \hline P & Q \end{array}\right)$, where $N \in \mathrm{GF}(q)^{k\times n-k}$, $P \in \mathrm{GF}(q)^{n-k\times k}$, $Q \in \mathrm{GF}(q)^{n-k\times n-k}$. We give a program for M in two main steps and no more than $\lfloor 3n/2 \rfloor$ instructions.

The first step computes $(M_S|N)$. By hypothesis, M_S can be computed in $\lfloor 3k/2 \rfloor$ instructions. We can easily convert that program in order to compute the matrix $(M_S|N)$ as follows. Consider the final update of row j: $y_j \leftarrow f_j$ (i.e., the j-th row must be equal to that of M after its last update). The j-th row of N, say n_j is a linear combination of the rows of $(0|I_{n-k})$, hence simply replace $y_j \leftarrow f_j$ by $y_j \leftarrow f_j + n_j$ and in any subsequent instruction, replace every occurrence of y_j by $y_j - n_j$.

The second step computes $(P|Q)$. Note that the rows $p_1,\ldots,p_{n-k}$ of P can be expressed as linear combinations of those of M_S: $P = RM_S$ where the rows of $R = PM_S^{-1} \in \mathrm{GF}(q)^{n-k\times k}$ are denoted $r_1,\ldots,r_{n-k}$. By hypothesis, the matrix $X := Q - RN$ (with rows $x_1,\ldots,x_{n-k}$) can be computed in $\lfloor 3(n-k)/2 \rfloor$ instructions. Again this can be converted to compute $(P|Q)$ as follows. Suppose i is the first row to have its last update in a program computing X, say it is $y_i \leftarrow \sum_{l=1}^{n-k} a_{i,l} y_l$. Then the new program for $(P|Q)$ is

$$y_{k+i} \leftarrow \sum_{l=k+1}^{n} a_{i,l} y_l + \sum_{l=1}^{k} r_{i,l} y_l = (r_i M_S | x_i + r_i N) = (p_i|q_i).$$

Then replace every future occurrence of y_i with $y_{k+i} - \sum_{l=1}^{k} r_{i,l} y_l$. Suppose that i' is the next row to have its last update $y_{i'} \leftarrow \sum_{l=1}^{n-k} a_{i',l} y_l$; this is converted to

$$y_{k+i'} \leftarrow \sum_{l=k+1}^{n} a_{i',l} y_l - a_{i',i} \sum_{l=1}^{k} r_{i,l} y_l + \sum_{l=1}^{k} r_{i',l} y_l = (r_{i'} M_S | x_{i'} + r_{i'} N) = (p_{i'}|q_{i'}).$$

Again, every future occurrence of i' will be replaced with $y_{k+i'} - \sum_{l=1}^{k} r_{i',l} y_l$, and so on. By induction, we can then easily prove that this converted program computes $(P|Q)$.

Now suppose M does not have any invertible principal. Let D be the directed graph whose adjacency matrix A_D satisfies $A_D(i,j) = 1$ if $M(i,j) \neq 0$ and $A_D(i,j) = 0$ if $M(i,j) = 0$. If D is acyclic, then M can be computed in n instructions, for it is (up to renaming the vertices in topological order) an upper triangular matrix with zeros on the diagonal. Otherwise, D has girth n, for otherwise the adjacency matrix of the subgraph induced by a shortest cycle forms a nonsingular principal. Therefore D is a cycle, and M can be computed in $n+1$ instructions by [12, Proposition 6].

The tightness of the bound follows from [12, Corollary 1]. In particular, the maximum is achieved for the permutation matrices corresponding to fixed-point free involutions for even n. □

By the proof of Theorem 2.1, we see that the only matrices in $\mathrm{GL}(2,q)$ which are a product of three instructions are exactly those whose support is the permutation matrix of a transposition. Proposition 2.2 below extends this result to any even order when the matrices are over $\mathrm{GF}(2)$.

Proposition 2.2. *In* $\mathrm{GL}(2m,2)$*, the only matrices which are the product of no fewer than* $3m$ *instructions are the permutation matrices of fix-point free involutions.*

Proof. We prove it by strong induction on m; it is clear for $m = 1$ and checked by computer for $m = 2$, therefore we assume $m \geq 3$ and that it holds for up to $m-1$. For any $k \geq 1$, we denote the permutation matrix of $(1,2)\cdots(2k-1,2k)$ as J_k. We say that two matrices M and N are equivalent if $M = \Pi N \Pi^{-1}$ for some permutation matrix Π.

Let $M \in \mathrm{GL}(2m,2)$ be a matrix at distance $3m$ from the identity which is not equivalent to J_m. According to the proof of Theorem 2.1, the graph D with adjacency matrix M must contain a directed cycle of length $< 2m$. The graph D has girth 2, for otherwise there is a principal of size other than 2 and hence M can be computed in fewer than $3m$ instructions by using the two-step algorithm in the proof of Theorem 2.1. More generally, any invertible principal of M must have even size and be a conjugate of J_k for some k.

Hence we can express M (up to equivalence) as $M = \left(\begin{array}{c|c} J_1 & N \\ \hline P & Q \end{array}\right)$, where $N \in \mathrm{GF}(2)^{2\times 2(m-1)}$, $P \in \mathrm{GF}(2)^{2(m-1)\times 2}$, $Q \in \mathrm{GF}(2)^{2(m-1)\times 2(m-1)}$. By the same argument, we can first compute J_1 and then the matrix $Q + PJ_1N$, hence these matrices must satisfy (up to equivalence) $PJ_1N + Q = J_{m-1}$.

Since $M \neq J_m$, there exists $2 \leq k \leq m$ such that the $\{1,2,2k-1,2k\}$-principal of M is not equal to J_2. The conditions above mean that this principal is not invertible, neither is any of its T-principals for $|T| = 3$, and it can be expressed as

$$\begin{pmatrix} 0 & 1 & a & b \\ 1 & 0 & c & d \\ e & f & 0 & \alpha \\ g & h & \beta & 0 \end{pmatrix},$$

where $\alpha = bf + de + 1$ and $\beta = ah + cg + 1$. However, it can be verified that no such matrix exists. □

We remark that the situation for $\mathrm{GL}(2m+1,2)$ is much more complicated. Indeed, the permutation matrices of $(1,2)(3,4)\cdots(2m-1,2m,2m+1)$ and its conjugates are still extremal, but many other matrices are also extremal. For example by Theorem 2 we know that the diameter of the Cayley graph for $\mathrm{GL}(3,2)$ is 4 and clearly there are only two extremal permutation matrices in $\mathrm{GL}(3,2)$ however there are 35 matrices equal to the product of 4 and no fewer linear instructions in this group - see Table 1.

3 Some matrix groups

We first discuss the special linear groups. Recall that a *transvection* is any linear permutation $t_{\phi,v}$ of $\mathrm{GF}(q)^n$, such that

$$t_{\phi,v}(x) = x + x\phi^\top v$$

for all $x \in \mathrm{GF}(q)^n$, where $v, \phi \in \mathrm{GF}(q)^n$ are nonzero and satisfy $v\phi^\top = 0$, [10]. By a suitable change of basis, any transvection can be represented by a shear matrix $S(i, e^i + ae^j)$ for some i, j and $a \in \mathrm{GF}(q)$; i.e. it is an instruction.

Proposition 3.1. *(i) The group* $\mathrm{SL}(n,q)$ *is internally computable for any n and prime power q.*

(ii) If $q \neq 2$ *then* $\mathrm{SL}(n,q)$ *is not fast in* $\mathrm{GL}(n,q)$.

$$\begin{pmatrix}0&1&0\\0&0&1\\1&0&0\end{pmatrix} \quad \begin{pmatrix}0&0&1\\1&0&0\\0&1&0\end{pmatrix}$$

$$\begin{pmatrix}0&1&0\\1&0&0\\1&0&1\end{pmatrix} \quad \begin{pmatrix}0&1&0\\1&0&0\\0&1&1\end{pmatrix} \quad \begin{pmatrix}1&0&1\\0&0&1\\0&1&0\end{pmatrix} \quad \begin{pmatrix}1&1&0\\0&0&1\\0&1&0\end{pmatrix} \quad \begin{pmatrix}0&0&1\\0&1&1\\1&0&0\end{pmatrix} \quad \begin{pmatrix}0&0&1\\1&1&0\\1&0&0\end{pmatrix}$$

$$\begin{pmatrix}0&1&1\\1&0&0\\0&1&0\end{pmatrix} \quad \begin{pmatrix}0&1&1\\0&0&1\\1&0&0\end{pmatrix} \quad \begin{pmatrix}0&0&1\\1&0&0\\1&1&0\end{pmatrix} \quad \begin{pmatrix}0&0&1\\1&0&1\\0&1&0\end{pmatrix} \quad \begin{pmatrix}0&0&1\\0&1&0\\1&1&0\end{pmatrix} \quad \begin{pmatrix}0&1&0\\1&0&1\\1&0&0\end{pmatrix}$$

$$\begin{pmatrix}0&1&0\\0&1&1\\1&0&0\end{pmatrix} \quad \begin{pmatrix}0&1&0\\0&0&1\\1&0&1\end{pmatrix} \quad \begin{pmatrix}1&0&1\\1&0&0\\0&1&0\end{pmatrix} \quad \begin{pmatrix}0&0&1\\1&1&0\\0&1&0\end{pmatrix} \quad \begin{pmatrix}1&1&0\\0&0&1\\1&0&0\end{pmatrix} \quad \begin{pmatrix}0&0&1\\1&0&0\\0&1&1\end{pmatrix}$$

$$\begin{pmatrix}0&0&1\\1&0&1\\1&1&0\end{pmatrix} \quad \begin{pmatrix}0&1&0\\1&0&1\\1&1&0\end{pmatrix} \quad \begin{pmatrix}0&1&1\\0&0&1\\1&1&0\end{pmatrix} \quad \begin{pmatrix}0&1&1\\1&0&0\\1&1&0\end{pmatrix} \quad \begin{pmatrix}0&1&1\\1&0&1\\1&0&0\end{pmatrix} \quad \begin{pmatrix}0&1&1\\1&0&1\\0&1&0\end{pmatrix}$$

$$\begin{pmatrix}1&1&1\\0&0&1\\0&1&0\end{pmatrix} \quad \begin{pmatrix}0&1&0\\1&0&0\\1&1&1\end{pmatrix} \quad \begin{pmatrix}0&0&1\\1&1&1\\1&0&0\end{pmatrix}$$

$$\begin{pmatrix}1&1&0\\1&1&1\\1&0&0\end{pmatrix} \quad \begin{pmatrix}0&0&1\\1&1&1\\0&1&1\end{pmatrix} \quad \begin{pmatrix}1&1&1\\1&1&0\\0&1&0\end{pmatrix} \quad \begin{pmatrix}1&1&1\\0&0&1\\1&0&1\end{pmatrix} \quad \begin{pmatrix}1&0&1\\1&0&0\\1&1&1\end{pmatrix} \quad \begin{pmatrix}0&1&0\\0&1&1\\1&1&1\end{pmatrix}$$

Table 1: The 35 matrices in GL(3,2) that are a product of four linear instructions and their orbits under the action of Sym(3).

Proof. (*i*) This is simply the observation that any transvection is an instruction and the transvections are well known to generate the special linear group - see for instance [16, p.45].

(*ii*) We prove this in the case $n = 2$, the extension to the general case being clear. If $q \neq 2$ then there exists an element $\alpha \in \mathrm{GF}(q)$ such that $\alpha \neq 0, 1$. Inside $\mathrm{GL}(2,q)$ we thus have

$$\begin{pmatrix} \alpha & 0 \\ 0 & \alpha^{-1} \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \alpha^{-1} \end{pmatrix}$$

which expresses the above element of $\mathrm{SL}(2,q)$ as a product of two instructions. Inside $\mathrm{SL}(2,q)$ however we have that

$$\begin{pmatrix} 1+xy & x \\ y & 1 \end{pmatrix} = \begin{pmatrix} 1 & x \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ y & 1 \end{pmatrix},$$
$$\begin{pmatrix} 1 & x \\ y & 1+xy \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ y & 1 \end{pmatrix} \begin{pmatrix} 1 & x \\ 0 & 1 \end{pmatrix}$$

for any $x, y \in \mathrm{GF}(q)$. Since $\alpha \neq 1$ the original matrix cannot be of this form and thus cannot be expressed as a product of just two instructions inside $\mathrm{SL}(2,q)$. □

The argument in the proof of (*ii*) can be easily generalised to show that any subgroup of GL defined as the set of matrices with determinant in a proper subgroup of the multiplicative group of $\mathrm{GF}(q)$ is not fast.

We remark that if $q = 2$ then $\mathrm{SL}(n,q) = \mathrm{GL}(n,q)$. Unfortunately most other groups that are naturally matrix groups are not internally computable in their natural $\mathrm{GF}(q)$ modules.

Proposition 3.2. *Orthogonal groups of type +, unitary and symplectic groups are not internally computable.*

Proof. In the orthogonal and unitary cases this is simply the observation that a matrix A is an element of these groups if it satisfies $AA^\top = I$ or $A\bar{A}^\top = I$, respectively, where the bar indicates the automorphism of $\mathrm{GF}(q)$ of order 2 when it exists [16, p.66 & p.70]. Clearly no instruction satisfies either condition and so these groups contain no instructions whatsoever.

Elements of the symplectic group $\mathrm{Sp}(2n,q)$ are precisely the invertible matrices of the form $\left(\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right)$ where A, B, C and D are $n \times n$ matrices such that

$$\begin{aligned} AD^\top - BC^\top &= I, \\ AB^\top &= A^\top B \quad \text{and} \\ CD^\top &= C^\top D. \end{aligned}$$

For an instruction to be of the above form one of B or C must be the all zeros matrix and $A = D = I$. If $C = 0$, we see that B must be a matrix with only one nonzero entry, which lies on the diagonal; if $B = 0$, we obtain its transpose. Therefore, the symplectic instructions generate a group of matrices where A,B, C and D are all diagonal; this is clearly a proper subgroup of $\mathrm{Sp}(2n,q)$. □

Proposition 3.3. *The groups ${}^2B_2(2^{2r+1})$, ${}^3D_4(q)$, $G_2(q)$, ${}^2G_2(3^{2r+1})$ and ${}^2F_4(2^{2r+1})$ are not internally computable.*

Proof. We prove this in the case of ${}^2B_2(2^{2r+1})$ acting on its natural 4 dimensional $\mathrm{GF}(2^{2r+1})$ module the cases of ${}^2G_2(3^{2r+1})$ acting on its natural 7 dimensional $\mathrm{GF}(3^{2r+1})$ module and ${}^2F_4(2^{2r+1})$ acting on its natural 26 dimensional $\mathrm{GF}(2^{2r+1})$ module being entirely analogous. Furthermore analogous arguments apply to ${}^3D_4(q)$ and $G_2(q)$ acting on their natural 26 and 8 dimensional $\mathrm{GF}(q)$ modules respectively.

An instruction whose only non-zero off-diagonal entries are contained entirely on the bottom row must be contained in the subgroup of lower triangular matrices. The non-trivial elements of this subgroup, however, are of the form

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ \alpha\beta^{-1} & 1 & 0 & 0 \\ \alpha\beta & \beta^2 & 1 & 0 \\ \alpha^2 & 0 & \alpha\beta^{-1} & 1 \end{pmatrix}$$

where $\alpha \in \mathrm{GF}(2^{2r+1})$ and $\beta = \alpha^{2^{r+1}-1}$ [16, p.115]. Clearly this subgroup contains no instructions and so the subgroup of ${}^2B_2(2^{2r+1})$ generated by any instructions is a proper subgroup. □

4 Generating linear groups

The purpose of this section is to determine the minimum number of instructions sufficient to generate some matrix groups. The reader is reminded of the elements $S(i,v)$ that we defined just before Theorem 2.1. We also define the vectors $v^i \in \mathrm{GF}(q)^n$ such that $v^i = e^i + e^{i+1}$ for $i \leq n-1$ and $v^n = e^1 + e^n$.

We first consider the special linear group.

Theorem 4.1. *The group $\mathrm{SL}(n,q)$ is generated by n instructions unless $n = 2$, $q = 2^m$ ($m \geq 2$), where it is generated by 3 instructions.*

Proof. The rest of the proof goes by induction on n, but we split the proof according to the parity of q. First, suppose q is odd. An immediate consequence of a classical Theorem incorrectly attributed to Dickson [11] (it was actually proved by Wilman and Moore, see [13, Corollary 2.2]) tells us that the maximal subgroups of $\mathrm{PSL}(2,q)$, q odd (these can easily be seen to "lift" to maximal subgroups of $\mathrm{SL}(2,q)$) are all isomorphic to one of

- $\mathrm{Alt}(4)$, $\mathrm{Sym}(4)$ or $\mathrm{Alt}(5)$;
- A dihedral group of order either $q+1$ or $q-1$;
- A subfield subgroup;
- A stabiliser of a one dimensional subspace in the action on the $q+1$ subspaces of $\mathrm{GF}(q)^2$ on which $(\mathrm{P})\mathrm{SL}(2,q)$ naturally acts.

Consider the matrices/instructions

$$S(1,(1,x)) = \begin{pmatrix} 1 & x \\ 0 & 1 \end{pmatrix}, \quad S(2,(y,1)) = \begin{pmatrix} 1 & 0 \\ y & 1 \end{pmatrix}.$$

We prove that the group they generate does not belong to any of the maximal subgroups. First, the copies of Alt(4), Sym(4) and Alt(5). In characteristic 3, the only way two elements of order 3 can be contained in a copy of Alt(4) or Sym(4) is if their product has order 1 or 3 (in which case they're contained in the same cyclic subgroup, which the above two matrices clearly are not) or 2 (and by direct calculation our two matrices do not have a product of order 2). Finally we can eliminate Alt(5) since this subgroup can only exist in characteristic 3 if $q = 3$ or 9 which are easily eliminated by computer. In characteristic 5 there are no elements order 5 in Alt(4) and Sym(4) and for Alt(5) this maximal subgroup only exists when q satisfies certain congruences that a power of 5 never satisfies. For characteristic greater than $p > 5$ there are clearly no elements of order p in any of Alt(4), Sym(4) or Alt(5).

Since p is coprime to both $q+1$ and $q-1$ neither of these belong to a maximal dihedral subgroup. The only one dimensional subspace fixed by the first matrix is spanned by the (column) vector $(1,0)^\top$ whilst the second only fixes the subspace spanned by the (column) vector $(0,1)^\top$, so no one dimensional subspace is fixed by the subgroup these generate. Recall that the product of these two matrices is the matrix $\begin{pmatrix} 1+xy & x \\ y & 1 \end{pmatrix}$ which has trace $2+xy$. Choosing x and y so that $2+xy$ is contained in no proper subfield of $\mathrm{GF}(q)$ now gives a pair of elements that cannot generate a subfield subgroup. It follows that this pair must generate the whole group.

We now prove the inductive step. Let $x, y \in \mathrm{GF}(q)$ such that the instructions $\begin{pmatrix} 1 & x \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ y & 1 \end{pmatrix}$ generate $\mathrm{SL}(2,q)$. Then we claim that the following set of n instructions generates $\mathrm{SL}(n,q)$:

$$\{S(i,v^i) : 1 \leq i \leq n-2\} \cup \{S(n-1, e^{n-1} + xe^n), S(n, e^n + ye^1)\}.$$

Let us remark that we can easily generate any instruction of the form $S(i, e^i + e^j)$ for $1 \leq i < j \leq n-1$ (and hence any of the form $S(i, e^i - e^j)$ as well). We can then easily generate $S(i, e^i + xe^n)$ for any $1 \leq i \leq n-1$. We also generate any transvection of the form $S(n, e^n + ye^i)$ for any $1 \leq i \leq n-1$ as such:

$$S(n, e^n + ye^i) = S(n, e^n - ye^1) S(1, e^1 - e^i) S(n, e^n + ye^1) S(1, e^1 + e^i).$$

Displaying only the columns and rows indexed $1, i, n$, the equation above reads

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & y & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -y & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ y & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

By combining the two types of transvections, we obtain all possible transvections of the type $S(i, e^i + ae^n)$ or $S(n, e^n + ae^i)$ for all $a \in \mathrm{GF}(q)$. We are done with the last coordinate, and we tackle the penultimate coordinate by considering

$$Q = \left(\begin{array}{c|cc} I_{n-2} & \multicolumn{2}{c}{0} \\ \hline 0 & 0 & -1 \\ & 1 & 0 \end{array}\right).$$

Note that Q is indeed generated by $S(n-1, e^{n-1} + xe^n)$ and $S(n, e^n + ye^{n-1})$. We then obtain the two

required types of transvections:

$$S(n-1, e^{n-1} + ye^i) = QS(n, e^n + ye^i)Q^{-1}$$
$$S(i, e^i + xe^{n-1}) = QS(i, e^i + xe^n)Q^{-1}.$$

The proof goes on from $n-1$ down to 2, thus generating any possible transvection.

Now suppose q is even. Any instruction in $\mathrm{SL}(2,2^m)$ is an element of order two, and hence any group generated by two instructions is dihedral. However, $\mathrm{SL}(2,2^m)$ is not a dihedral group for $m \geq 2$ and hence cannot be generated by two instructions. We now prove it can be generated by three instructions.

We recall from Dickson's theorem [11] that the maximal subgroups of $\mathrm{SL}(2,2^m)$ are each isomorphic to either

- a stabiliser of a one dimensional subspace in the action on the 2^m+1 subspaces of $\mathrm{GF}(2^m)^2$ on which $\mathrm{(P)SL}(2,2^m)$ naturally acts;
- a subfield subgroup;
- a dihedral group of order $2(2^m \pm 1)$.

Consider the matrices

$$A := \begin{pmatrix} 1 & 0 \\ x & 1 \end{pmatrix}, \quad B := \begin{pmatrix} 1 & x \\ 0 & 1 \end{pmatrix}, \quad C := \begin{pmatrix} 1 & x^2 \\ 0 & 1 \end{pmatrix}$$

where $x \in \mathrm{GF}(2^m)$ is contained in no proper subfield. Let H be the subgroup generated by the matrices A and B. By the same arguments as the case $\mathrm{SL}(2,q)$ with q odd we know that H is contained in neither a subspace stabilizer nor a subfield subgroup and so the only maximal subgroups containing H must be dihedral of order $2(q \pm 1)$. Note that since these are dihedral groups of twice odd order these subgroups cannot contain pairs of involutions that commute. Since $BC = CB$ it follows that C cannot be contained in any of these dihedral subgroups and so no maximal subgroup contains all of A, B and C, hence they must generate the whole group.

The base case of the induction thus occurs for $n = 3$. Let x such that $\begin{pmatrix} 1 & 0 \\ x & 1 \end{pmatrix}$, $\begin{pmatrix} 1 & x \\ 0 & 1 \end{pmatrix}$ and $\begin{pmatrix} 1 & x^2 \\ 0 & 1 \end{pmatrix}$ generate $\mathrm{SL}(2,2^m)$. We shall prove that the matrices

$$M_1 := \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, M_2 := \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & x \\ 0 & 0 & 1 \end{pmatrix}, M_3 := \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x & 0 & 1 \end{pmatrix}$$

generate $\mathrm{SL}(3,2^m)$. Denoting

$$N_1 := M_1^{-1}M_2^{-1}M_1M_2 = \begin{pmatrix} 1 & 0 & x \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad N_2 := M_2^{-1}M_3^{-1}M_2M_3 = \begin{pmatrix} 1 & 0 & 0 \\ x^2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

we obtain

$$P_3 := N_2^{-1}N_1^{-1}N_2N_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & x^3 \\ 0 & 0 & 1 \end{pmatrix}.$$

Since

$$P_3^{-1}M_3^{-1}P_3M_3 = \begin{pmatrix} 1 & 0 & 0 \\ x^4 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

we can proceed as above to obtain $S(2,(0,1,x^5))$. We may repeat this process until we derive $S(2,(0,1,x^{2m+1})) = S(2,(0,1,x^2))$, which together with M_2 and

$$M_3^{-1}M_1^{-1}M_3M_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & x & 1 \end{pmatrix}$$

generate $\mathrm{SL}(2,2^m)$ acting on the last two coordinates. It is then easy to show that any transvection of the form $S(1,e^1+ae^i)$ or $S(i,e^i+ae^1)$ for any $i=2,3$ and any $a \in \mathrm{GF}(2^m)$ can be generated. Thus, the whole special linear group is generated.

We now prove the inductive step. More specifically, we show that $\mathrm{SL}(n,q)$ is generated by the following set of instructions:

$$\{S(i,v^i) : 1 \le i \le n-2\} \cup \{S(n-1,e^{n-1}+xe^n), S(n,e^n+xe^1)\}.$$

Again, we can easily generate $S(1,e^1+xe^n)$ and hence $\mathrm{SL}(3,2^m)$ acting on the coordinates 1, $n-1$, and n. In particular, $S(n-1,e^{n-1}+xe^1)$ is generated and by induction hypothesis we obtain $\mathrm{SL}(n-1,2^m)$ acting on the first $n-1$ coordinates. Finally, any transvection of the form $S(n,e^n+ae^i)$ or $S(i,e^i+ae^n)$ for any $i \le n-1$ and any $a \in \mathrm{GF}(2^m)$ can be easily generated. Thus, the whole special linear group is generated. □

We now turn to the general linear group.

Theorem 4.2. *The group* $\mathrm{GL}(n,q)$ *is generated by n instructions for any n and any prime power q.*

Proof. The proof is split into two parts, depending on the parity of q; the even part goes by induction on n. If q is even, we prove that $\mathrm{GL}(n,2^m)$ is generated by the n instructions

$$\{S(i,v^i) : 2 \le i \le n-1\} \cup \{S(1,\alpha e^1+e^2), S(n,\alpha e^1+e^n)\}$$

for any primitive element α. Since $\det(S(1,\alpha e^1+e^n)) = \alpha$, we only need to generate the special linear group.

For $n=2$, denote $M_i = S(i,(\alpha,1))$ for $i=1,2$. Then we can generate the transposition matrix as follows: $P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = M_1M_2M_1^{-1}$. Since $S(1,(1,\alpha)) = PM_2P$, we easily generate $S(1,(\alpha,0)) = M_1^{-1}S(1,(1,\alpha))M_1^2$. Any transvection $S(1,(1,\alpha^k))$ can then be expressed as

$$\begin{pmatrix} 1 & \alpha^k \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \alpha^{k-1} & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha^{-k+1} & 0 \\ 0 & 1 \end{pmatrix},$$

and any other transvection is obtained by conjugating by P.

We now prove the inductive part. We can easily generate $S(1, \alpha e^1 + e^n)$, which combined with $S(n, e^n + \alpha e^1)$ generates $\mathrm{GL}(2,q)$ acting on the coordinates 1 and n. In particular, we obtain the matrix $Q = \begin{pmatrix} 0 & 1 \\ \alpha & 0 \end{pmatrix}$, and

$$S(n-1, \alpha e^1 + e^{n-1}) = Q^{-1} S(n, \alpha e^1 + e^n) Q.$$

We then have the complete set of generators for $\mathrm{GL}(n-1,q)$ acting on coordinates 1 to $n-1$. It is then easy to prove that any transvection of the form $S(i, e^i + ae^n)$ and $S(n, e^n + ae^i)$ for any $1 \leq i \leq n-1$ and any $a \in \mathrm{GF}(q)$ can be generated.

If q is odd and $n = 2$, consider the matrices $A := \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, $B := \begin{pmatrix} 1 & 0 \\ 1 & x \end{pmatrix}$ where $x \in \mathrm{GF}(q)$ is not contained in any proper subfield. Arguments analogous to those used in the $\mathrm{SL}(2,q)$ case show that $\langle A, A^B \rangle = \mathrm{SL}(2,q)$.

If $n > 2$, we rely on the proof of Theorem 4.1 for q odd. We know that there exist $x, y \in \mathrm{GF}(q)$ such that $\mathrm{SL}(n,q)$ is generated by

$$\{S(i, v^i) : 1 \leq i \leq n-2\} \cup \{S(n-1, e^{n-1} + xe^n), S(n, e^n + ye^1)\}.$$

Let a be a primitive element of $\mathrm{GF}(q)$ and $b := (a-1)x/2$. We shall prove that replacing the instruction updating coordinate $n-1$ by $T = S(n-1, ae^{n-1} + be^n)$ in the set above yields a generating set for $\mathrm{GL}(n,q)$. We only need to show that $S(n-1, e^{n-1} + xe^n)$ is generated. We have $T^{(q-1)/2} = S(n-1, -e^{n-1} - xe^n)$ and hence we can easily generate $S(1, e^1 + xe^n)$ and the whole of $\mathrm{SL}(2,q)$ acting on coordinates 1 and n. In particular, we obtain $Q = \mathrm{diag}(2^{-1}, 1, \ldots, 1, 2)$, whence

$$S(n-1, e^{n-1} + xe^n) = S(n-1, -e^{n-1} - xe^n) Q^{-1} S(n-1, -e^{n-1} - xe^n) Q.$$

Only displaying rows and columns indexed $1, n-1, n$ the equation above reads

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & x \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & -x \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2^{-1} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & -x \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2^{-1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}.$$

□

We conclude this section by noticing that Theorems 4.1 and 4.2 have implication on some classical semigroups of matrices. Denote the semigroup of singular matrices in $\mathrm{GF}(q)^{n \times n}$ as $\mathrm{Sing}(n,q)$ and consider the general linear semigroup (also called full linear monoid [14]) and special linear semigroup:

$$\mathrm{GLS}(n,q) = \mathrm{GL}(n,q) \cup \mathrm{Sing}(n,q),$$
$$\mathrm{SLS}(n,q) = \mathrm{SL}(n,q) \cup \mathrm{Sing}(n,q).$$

Note that $\mathrm{Sing}(n,q)$ is not an internally computable semigroup. Indeed, the kernel of any singular instruction matrix only contains vectors with Hamming weight equal to zero or one. Thus any matrix

whose kernel forms a code with minimum distance at least two cannot be computed by a program only consisting of singular instructions. For instance, the square all-ones matrix of any order over any finite field cannot be computed in that fashion.

However, according to Theorems 6.3 and 6.4 in [15], any generating set of $\mathrm{GL}(n,q)$ ($\mathrm{SL}(n,q)$ respectively) appended with any matrix of rank $n-1$ in $\mathrm{Sing}(n,q)$ generates $\mathrm{GLS}(n,q)$ ($\mathrm{SLS}(n,q)$ respectively). Since any singular instruction has rank $n-1$, we conclude that these semigroups are internally computable, and in particular $\mathrm{GLS}(n,q)$ is generated by $n+1$ instructions, while $\mathrm{SLS}(n,q)$ is generated by $n+1$ instructions unless $q=2^m$ and $n=2$, where it is generated by four instructions.

References

References

[1] RUDOLF AHLSWEDE, NING CAI, SHUO-YEN ROBERT LI, AND RAYMOND W. YEUNG: Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000. 2

[2] SERGE BURCKEL: Closed iterative calculus. *Theoretical Computer Science*, 158:371–378, May 1996. 2

[3] SERGE BURCKEL: Elementary decompositions of arbitrary maps over finite sets. *Journal of Symbolic Computation*, 37(3):305–310, 2004. 2

[4] SERGE BURCKEL, EMERIC GIOAN, AND EMMANUEL THOMÉ: Mapping computation with no memory. In *Proc. International Conference on Unconventional Computation*, pp. 85–97, Ponta Delgada, Portugal, September 2009. 2

[5] SERGE BURCKEL, EMERIC GIOAN, AND EMMANUEL THOMÉ: Computation with no memory, and rearrangeable multicast networks. *Discrete Mathematics and Theoretical Computer Science*, 16:121–142, 2014. 2, 4

[6] SERGE BURCKEL AND MARIANNE MORILLON: Three generators for minimal writing-space computations. *Theoretical Informatics and Applications*, 34:131–138, 2000. 2

[7] SERGE BURCKEL AND MARIANNE MORILLON: Quadratic sequential computations of boolean mappings. *Theory of Computing Systems*, 37(4):519–525, 2004. 2

[8] SERGE BURCKEL AND MARIANNE MORILLON: Sequential computation of linear boolean mappings. *Theoretical Computer Science*, 314:287–292, February 2004. 2

[9] PETER J. CAMERON, BEN FAIRBAIRN, AND MAXIMILIEN GADOULEAU: Computing in permutation groups without memory. *Chicago Journal of Theoretical Computer Science*, to appear. 4

[10] PETER J. CAMERON AND JOHN HALL: Some groups generated by transvection subgroups. *Journal of Algebra*, 140(1):184–209, June 1991. 6

[11] L.E. DICKSON: *Linear groups, with an Exposition of the Galois Field Theory*. Teubner, Leipzig, 1901. 9, 11

[12] MAXIMILIEN GADOULEAU AND SØREN RIIS: Memoryless computation: new results, constructions, and extensions. *Theoretical Computer Science*, To appear. Available at http://www.sciencedirect.com/science/article/pii/S0304397514007300. 2, 3, 4, 5

[13] O. H. KING: The subgroup structure of finite classical groups in terms of geometric configurations. In B.S. WEBB, editor, *Surveys in Combinatorics, 2005*, pp. 29–56. Cambridge University Press, 2006. 9

[14] J. OKNINSKI: *Semigroups of Matrices*. World Scientific Publishing, 1998. 13

[15] NIKOLA RUŠKUC: *Semigroup Presentations*. Ph. D. thesis, University of St Andrews, 1995. 14

[16] ROBERT A. WILSON: *The Finite Simple Groups*. Springer, 2009. 8, 9

[17] RAYMOND W. YEUNG, SHUO-YEN ROBERT LI, NING CAI, AND ZHEN ZHANG: *Network Coding Theory*. Volume 2 of *Foundation and Trends in Communications and Information Theory*. now Publishers, Hanover, MA, 2006. 2

AUTHORS

Peter J. Cameron
Professor of Mathematics and Statistics
School of Mathematics and Statistics
University of St Andrews, UK
pjc20@st-andrews.ac.uk
http://www-circa.mcs.st-andrews.ac.uk/~pjc/
Emeritus Professor of Mathematics
School of Mathematical Sciences
Queen Mary, University of London, UK
p.j.cameron@qmul.ac.uk
http://www.maths.qmul.ac.uk/~pjc/

Ben Fairbairn
Lecturer in Mathematics
Department of Economics, Mathematics and Statistics
Birkbeck, University of London, UK
b.fairbairn@bbk.ac.uk
http://www.bbk.ac.uk/ems/faculty/fairbairn

Maximilien Gadouleau
Lecturer in Computer Science
School of Engineering and Computing Sciences
Durham University, Durham, UK
m.r.gadouleau@durham.ac.uk
http://community.dur.ac.uk/m.r.gadouleau/